UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/870,624	05/31/2001	Scott J. Broussard	AUS920010268US1	1775

35617 7590 10/07/2005

DAFFER MCDANEIL LLP
P.O. BOX 684908
AUSTIN, TX 78768

EXAMINER	
BONSHOCK, DENNIS G	

ART UNIT	PAPER NUMBER
2173	

DATE MAILED: 10/07/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/870,624
Filing Date: May 31, 2001
Appellant(s): BROUSSARD, SCOTT J.

Kevin L. Daffer (reg. 34,146)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 7-25-2005.

PT

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

The following are the related appeals, interferences, and judicial proceedings known to the examiner which may be related to, directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal:

09/870,613: Notice of Appeal filed 2/7/05; Appeal Brief filed on or about April 7, 2005.

09/870,614: Notice of Appeal filed 10/26/04; Appeal Brief filed on or about December 20, 2004.

09/870,615: Notice of Appeal filed 9/14/04; Appeal Brief filed on or about November 9 2004.

09/870,620: Notice of Appeal filed 12/7/04; Appeal Brief filed on or about February 7, 2005.

09/870,621: Notice of Appeal filed 9/24/04; Appeal Brief filed on or about November 23, 2004.

09/870,622: Notice of Appeal filed 8/24/04; Appeal Brief filed on or about October 25, 2004.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

No evidence is relied upon by the examiner in the rejection of the claims under appeal.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nason, Patent #6,727,918, and Fowler, "Mixing heavy and light components."

3. With regard to claim 1, Nason teaches a display (see column 5, lines 33-44), a graphical user interface (see column 5, lines 18-21), a processor for implementing the embodiments of the invention (see claim 48), the system being implemented on a windows based operating system (see column 2, lines 46-50), implementing two different APIs, one independent of OS, one dependent on OS, to generate images (see column 5, lines 18-22 and lines 45-63), and a reading and rewriting of screen display information, where the primary GUI information is maintained by replacing the primary GUI with a secondary GUI (see column 25, lines 27-40). Nason also teaches the use of Java, but doesn't get in to the specifics. Fowler teaches a system of mixing two APIs

similar to that of Nason (see page 1, paragraphs 1 and 2), but further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on page 7, the components having the same look and feel. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs in Java, which is taught by Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

4. With regard to claim 2, which teaches the first and second images having the same look and feel, Fowler further teaches, on page 7, the components having the same look and feel.

5. With regard to claims 3 and 11, which teach the first and second images comprising pixels presented upon the display via the graphical user interface associated with the application program, Nason further teaches, in column 6, lines 14-37, the use of pixels for presenting the image on the display.

6. With regard to claims 4 and 12, which teach the first and second images comprise images of an object selected from a group comprising buttons, list boxes, and slide bars on which a pointer device can be directed by a user, Nason teaches, in column 6, lines 25-30, the display not being limited to, buttons, menus, application output controls animations, and use input controls. Further more, Fowler teaches, on page 1, paragraph 1, Swing an AWT containing components such as buttons, lists, and the like.

7. With regard to claims 5 and 13, which teach an application program written in Java programming language, Nason further teaches, in column 5, lines 60-63, the content controller including content and operating software such as JAVA.

8. With regard to claims 6 and 14, which teach the software component comprising a java application program interface consisting of an abstract windowing toolkit (AWT) during a second time, Nason further teaches, the use Java in the system. Fowler teaches a system in which two APIs can be used, similar to that of Nason, but also teaches the use of Swing and AWT in the same application program (see page 1, paragraph 2). It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs used in Java, which is also mentioned in Nason (see column 1, line 39 and column 10, line 40). Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1,

paragraph 2), would add an element of platform independence to the invention of Nason.

9. With regard to claims 7 and 15, which teach the software component comprising a java application program interface consisting of an Swing application program interface during a first time, Nason further teaches, the use Java in the system. Fowler teaches a system in which two APIs can be used, similar to that of Nason, but also teaches the user of Swing and AWT in the same application program (see page 1, paragraph 2). It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs used in Java, which is also mentioned in Nason (see column 1, line 39 and column 10, line 40). Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

10. With regard to claims 8 and 16, which teach the operating system comprising a Windows, Unix, or OS/2 computer operating system, Nason further teaches, in column 2, lines 46-50, the use of operating systems such as Windows, Linux, Apple's Macintosh OS/2, or Unix.

11. With regard to claims 9 and 17, which teach the first and second images presenting the same look and feel upon the display independent of the operating

system, Fowler further teaches, on page 2, that the java look and feel, that of AWT and Swing, that provides a distinctive platform-independent look and feel.

12. With regard to claim 10, Nason teaches implementing two different APIs, one independent of OS and one dependent on OS, to generate images (see column 5, lines 18-22 and lines 45-63), and a reading and rewriting of screen display information, where the primary GUI information is maintained by replacing the primary GUI with a secondary GUI (see column 25, lines 27-40). Nason also teaches the use of Java, but doesn't get in to the specifics. Fowler teaches a system of mixing two APIs similar to that of Nason (see page 1, paragraphs 1 and 2), but further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on page 7, the components having the same look and feel. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs in Java, which is taught by Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

13. With regard to claim 18, Nason teaches a computer-readable storage device, comprising: an operating system (see column 2, lines 46-50), and application program adapted for executing code of a software component (ex: APIs) (see column 3, lines 36-39 and column 5, lines 45-64), implementing two different APIs to generate objects in the same application program (see column 5, lines 45-64), and a reading and rewriting of screen display information, where the primary GUI information is maintained by replacing the primary GUI with a secondary GUI (see column 25, lines 27-40). Nason also teaches the use of two Java APIs, but doesn't get in to the specifics of each of the APIs. Fowler teaches a system of mixing two APIs similar to that of Nason (see page 1, paragraphs 1 and 2), but further teaches specifics of AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on page 7, the components having the same look and feel. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made that the first image would overwrite an image upon the display previous to the first image and that the first image can't overwrite the second image during the first time (because it hasn't been generated yet) and also to modify the system of using two APIs in one application program, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because the APIs of Java are use in

Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

14. With regard to claim 19, which teach the software component comprising a java application program interface consisting of a Swing application program interface during a first time, Nason further teaches, the use Java in the system for complementary user interfaces coexisting, with a primary interface. Fowler teaches a system in which two APIs can be used, similar to that of Nason, but also teaches the user of Swing and AWT in the same application program (see page 1, paragraph 2). It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs used in Java, which is also mentioned in Nason (see column 1, line 39 and column 10, line 40). Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

15. With regard to claim 20, which teaches the software component comprising a java application program interface consisting of an abstract windowing toolkit (AWT) during a second time, Nason further teaches, the use Java in the system for complementary user interfaces coexisting, with a primary interface. Fowler teaches a system in which two APIs can be used in the same application program similar to that of

Nason, but also teaches the use of Swing and AWT in the same application program (see page 1, paragraph 2). Fowler further teaches specifics of AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs implemented inn Java, which is claimed Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add an element of platform independence to the invention of Nason.

(10) Response to Argument

Claims 1-9:

With respect to the group of claims including Claims 1-9, the Appellant's arguments are focused on the limitations regarding the "image generated during a first time (e.g., the OS-independent image) and the image generated during the second time (e.g., the OS-dependent image) are substantially identical to one another." More specifically, as stated from representative Claim 1, the limitation argued is:

"the executed software component generates a first image upon the display independent of code within the operating system during a first time and, during a second time, emulates code that, when executed by the processor, generates a second image upon the display dependent on code within the operating system, and wherein the first and second images are substantially identical."

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to the creating of both a platform independent image and a platform dependent image upon a display where the two images are similar. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

"*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure.* In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the Nason and Fowler references are within the scope of these limitations.

Nason teaches a system for providing a second graphical interface with a primary graphical interface (see column 5, lines 18-22), a display (see column 5, lines

33-44), a memory (see column 3, line 65 though column 4, line 3), platform independent components for implementing the system (see column 5, lines 45-63), and implementing the system using content and operating software in JAVA (see column 5, lines 60-63). JAVA is known in the art to comprise GUIs such as AWT, a JAVA GUI that is known to rely on the native GUI of the computer on which the application runs, and SWING, a JAVA GUI that runs uniformly on any native platform (see Microsoft Computer Dictionary pages 13 and 505). Nason teaches an Alternate Display Content Controller (ADCC) that contains an Application Program Interface (API) that intercepts all calls to a graphics device interfaces (GDI) (native to the operating systems) (platform dependent display routines) and routes them to a complementary GUI display driver [see column 20, line 46 through column 21, line 28]. This complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). Nason teaches the use of Java, but doesn't get in to the specifics. Fowler teaches a system of mixing two APIs similar to that of Nason (see page 1, paragraphs 1 and 2), but further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on pages 5 and 7, the components looking similar to one another. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at

the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs in Java, which is taught by Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add a further element of platform independence to the invention of Nason, which has been shown in Nason column 5, lines 45-63.

The examiner will now address the individual arguments and statements made by Appellant.

From pages 6 of the Appeal Brief, from the second paragraph, the Appellant argues that "The presently claimed case not only describes the user of two different APIs for generating images but more importantly, describes a means for displaying an image with a platform-independent interface during a first time, and displaying a substantially identical image with a platform-dependent interface during a second time."

The examiner respectfully contends that Nason shows both a system that can display elements dependent on the native operating system (see column 5, lines 45-60) and further shows a complementary GUI display driver (see column 20, line 46 through column 21, line 28) and that this complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5,

line 4 and column 5, lines 45-63). The applications are shown to be combined with the kernel, which is further operating system independent. The existence of these two different application program interfaces (APIs) in the same system is a concept common to both the Nason and Fowler references. Where the fowler reference specifically shows a depiction of screen elements generated by the OS-dependent and OS-independent interfaces, with the elements being displayed in a manner where they appear very similar in appearance (see pages 4 and 6).

From pages 6 of the Appeal Brief, from the third paragraph, the Appellant argues that Nason fails to provide teaching for "the first and second images being displayed at times".

The examiner respectfully contends that the claim 1 as currently presented does not limit to displaying the element at different times where only one is displayed at a time, as is implied by the applicant's arguments. Though the Nason reference is believed to be capable of such a feature, Nason has been proven to display elements on separate sections of a display (see column 5, lines 10-60) where if one area is updated at a different time than the other the image displayed is generated at a different time separate from the first.

From pages 8 of the Appeal Brief, from the second paragraph, the Appellant argues that "Fowler cannot be combined with Nason to provide teaching or suggestion for the presently claimed component."

The examiner respectfully contends that Fowler teaches a system of mixing two APIs on the same display, similar to that of Nason (see page 1, paragraphs 1 and 2), and further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3), further similar to the alternate displays of Nason. Fowler further adds the teaching that the displayed components are similar in appearance (see pages 4-6).

From pages 10 of the Appeal Brief, from the second paragraph, the applicants' argue that Fowler teaches away from displaying an image with the AWT API during a first time, and displaying the same image with the Swing API during a second time.

In response, the examiner respectfully submits that though Fowler warns of problems that may be encountered when mixing AWT and Swing, Fowler teaches, on page 1, paragraph 2, the mixing of AWT and Swing in the same application program, where the teaching of display of the same image with a second API is taught by Nason, in column 25, lines 27-40, as shown supra. Fowler warns of problems in mixing but gives the case where mixing is necessary.

From page 11 of the Appeal Brief, from the fourth paragraph, the Appellant argues that "The Examiner has failed to adequately support and/or establish a prima facie ground of obviousness."

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, both references teach the use of a OS-independent interface and an OS-dependent interface used together in the same system. Nason teaches the content and operating software can include JAVA, where Fowler teaches which API components in JAVA implement these OS-independent and OS-dependent interfaces.

Claims 10-17:

With respect to the group of claims including Claims 10-17, the Appellant's arguments are focused on the limitations regarding the replacing of the interface with another interface, and the OS-independent and OS-dependent versions being used for displaying images associated with the same application program. More specifically, as stated from representative Claim 1, the limitation argued is:

> *"displaying a first image upon a display of the computer using the first interface; replacing the interface with a second interface that is substantially independent of the operating system yet emulates the*

behavior of at least a part of the first interface; re-running the application program; and re-displaying a second image upon the display of the computer using the second interface, wherein the second image has substantially the same look and feel as the first image."

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to the creating of both a platform independent image and a platform dependent image, where after running the first interface (platform dependent) the interface (not the image) is replaced with the second interface (platform independent), these two images being similar. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

"*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure*. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the Nason and Fowler references are within the scope of these limitations.

Nason teaches a system for providing a second graphical interface with a primary graphical interface (see column 5, lines 18-22), a display (see column 5, lines

33-44), a memory (see column 3, line 65 though column 4, line 3), platform independent components for implementing the system (see column 5, lines 45-63), and implementing the system using content and operating software in JAVA (see column 5, lines 60-63). JAVA is known in the art to comprise GUIs such as AWT, a JAVA GUI that is known to rely on the native GUI of the computer on which the application runs, and SWING, a JAVA GUI that runs uniformly on any native platform (see Microsoft Computer Dictionary pages 13 and 505). Nason teaches an Alternate Display Content Controller (ADCC) that contains an Application Program Interface (API) that intercepts all calls to a graphics device interfaces (GDI) (native to the operating systems) (platform dependent display routines) and routes them to a complementary GUI display driver [see column 20, line 46 through column 21, line 28]. This complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). Nason further teaches in column 20, line 46 through column 21, line 33, a means for communication between the application and the API once the display area has been created, such as a part processing by the complementary GUI before passing to the native graphics display driver. This same section further shows that when a command is forwarded to the controller the controller determines where to send the command for display, in a system using two different APIs, such as this one, the original interface will be replaced, as the destination interface, by a different interface for processing to the display. Nason teaches the use of Java, but doesn't get in to the specifics. Fowler teaches a system of mixing two APIs similar to that of Nason (see page 1, paragraphs 1 and 2), but further

teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on pages 5 and 7, the components looking similar to one another. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs in Java, which is taught by Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add a further element of platform independence to the invention of Nason, which has been shown in Nason column 5, lines 45-63.

The examiner will now address the individual arguments and statements made by Appellant.

From pages 13 of the Appeal Brief, from the third paragraph, the Appellant argues that "Nason does not teach the method steps of: running an application program to display a first image using a first OS-dependent interface; replacing the interface with a second interface that is independent of the operating system

and re-running the application program to re-display a second image upon the display of the computer using the second interface, wherein the second image using the second interface, where the second image is substantially identical to the first image."

The examiner respectfully contends that Nason shows both a system that can display elements dependent on the native operating system (see column 5, lines 45-60) and further shows a complementary GUI display driver (see column 20, line 46 through column 21, line 28) and that this complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). Nason further teaches in column 20, line 46 through column 21, line 33, a means for communication between the application and the API once the display area has been created, such as a part processing by the complementary GUI before passing to the native graphics display driver. This same section further shows that when a command is forwarded to the controller the controller determines where to send the command for display, in a system using two different APIs, such as this one, the original interface will be replaced, as the destination interface, by a different interface for processing to the display. This second image that is said to be "re-displayed" is nothing more than another image that can be displayed in conjunction with the first but that was generated using a different API.

From pages 13 of the Appeal Brief, from the fourth paragraph, the Appellant argues that "Nason cannot provide teaching for the method step of

"running an application program to display a first image using a first OS-dependent interface," because Nason does not teach or suggest that the primary GUI can be used for displaying images."

The examiner respectfully contends that Nason teaches, in column 5, lines 18-22, that complementary user interfaces may coexist with a primary user interface; and further teaches in column 5, lines 45-60 that the alternate display content controller (ADCC) can be dependent on the native operating system.

From pages 14 of the Appeal Brief, from the third paragraph, the Appellant argues that "Nason does not disclose that the ADCC may be implemented in OS-independent or OS-dependent versions of the ADCC may both be used for displaying images associated with the same application program."

The examiner respectfully contends that Nason shows both a system that can display elements dependent on the native operating system (see column 5, lines 45-60) and further shows a complementary GUI display driver (see column 20, line 46 through column 21, line 28) and that this complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). Nason specifically teaches, in column 21, lines 23-28, that the ADCC intercepts the call, and determines whether to forward it to a display driver enabled with the techniques of the present invention "and" or to a native display driver. The "and" signifying the ability to forward the application to two different display drivers. Nason further teaches in column 20, line 46 through column 21, line 33, a

means for communication between the application and the API once the display area has been created, such as a part processing by the complementary GUI before passing to the native graphics display driver. This same section further shows that when a command is forwarded to the controller the controller determines where to send the command for display, in a system using two different APIs, such as this one, the original interface will be replaced, as the destination interface, by a different interface for processing to the display. This second image that is said to be "re-displayed" is nothing more than another image that can be displayed in conjunction with the first but that was generated using a different API.

From page 15 of the Appeal Brief, from the first paragraph, the Appellant argues that "Fowler cannot be combined with Nason to provide teaching or suggestion for the presently claimed method."

The examiner respectfully contends that Fowler teaches a system of mixing two APIs on the same display, similar to that of Nason (see page 1, paragraphs 1 and 2), and further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3), further

similar to the alternate displays of Nason. Fowler further adds the teaching that the displayed components are similar in appearance (see pages 4-6).

From page 16 of the Appeal Brief, from the first paragraph, the Appellant argues that "The Examiner has failed to adequately support and/or establish a prima facie ground of obviousness."

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, both references teach the use of a OS-independent interface and an OS-dependent interface used together in the same system. Nason teaches the content and operating software can include JAVA, where Fowler teaches which API components in JAVA implement these OS-independent and an OS-dependent interfaces.

Claims 18-20:

With respect to the group of claims including Claims 18-20, the Appellant's arguments are focused on the limitations regarding the image generated during a first time (e.g., the OS-dependent image) and the image generated during the second time

(e.g., the OS-independent image) are substantially identical to one another. More specifically, as stated from representative Claim 1, the limitation argued is:

> *"during a first time, generates a first image dependent of cod executing within the operating system; and during a second time, generates a second image independent on cod executing within the operating system, wherein the second image is adapted to overwrite the first image upon a display screen during the second time, and wherein the first and second images are substantially identical."*

Since the interpretation of the limitation is the basis for the arguments, the Examiner's interpretation is now given. The claim, as interpreted by the examiner, pertains to the creating of both a platform dependent image and a platform independent image upon a display where the two images are similar, and image created by the platform independent driver being capable of overwriting the first image. As stated in the eighth paragraph of MPEP 2101[R2].II.C.,

> "*Office personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure*. In re Morris, 127 F.3d 1048, 1054-55, 44 USPQ2d 1023,1027-28 (Fed. Cir. 1997)."

Based on the interpretation of the claim limitations being argued, the Examiner will now explain how the teachings of the Nason and Fowler references are within the scope of these limitations.

Nason teaches a system for providing a second graphical interface with a primary graphical interface (see column 5, lines 18-22), a display (see column 5, lines 33-44), a memory (see column 3, line 65 though column 4, line 3), platform independent components for implementing the system (see column 5, lines 45-63), and implementing the system using content and operating software in JAVA (see column 5, lines 60-63). JAVA is known in the art to comprise GUIs such as AWT, a JAVA GUI that is known to rely on the native GUI of the computer on which the application runs, and SWING, a JAVA GUI that runs uniformly on any native platform (see Microsoft Computer Dictionary pages 13 and 505). Nason teaches an Alternate Display Content Controller (ADCC) that contains an Application Program Interface (API) that intercepts all calls to a graphics device interfaces (GDI) (native to the operating systems) (platform dependent display routines) and routes them to a complementary GUI display driver [see column 20, line 46 through column 21, line 28]. This complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). Nason further teaches in column 20, line 46 through column 21, line 33, a means for communication between the application and the API once the display area has been created, such as a part processing by the complementary GUI before passing to the native graphics display driver. This same section further shows that when a command is forwarded to the controller the controller determines where to send the command for display, in a system using two different APIs, such as this one, the interface will be original interface will be

replaced as the destination interface by a different interface for processing to the display. Nason specifically teaches, in column 21, lines 23-28, that the ADCC intercepts the call, and determines whether to forward it to a display driver enabled with the techniques of the present invention "and" or to a native display driver. The "and" signifying the ability to forward the application to two different display drivers. Nason teaches the use of Java, but doesn't get in to the specifics. Fowler teaches a system of mixing two APIs similar to that of Nason (see page 1, paragraphs 1 and 2), but further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3). Fowler further teaches, on pages 5 and 7, the components looking similar to one another. It would have been obvious to one of ordinary skill in the art, having the teachings of Nason and Fowler before him at the time the invention was made to modify the system of using two APIs, of Nason, to include the use of the AWT and Swing APIs, as did Fowler. One would have been motivated to make such a combination because AWT and Swing are two well-known APIs in Java, which is taught by Nason. Also the use of the combination of AWT and Swing, which have been proven to be usable together (see Fowler page 1, paragraph 2), would add a further element of platform independence to the invention of Nason, which has been shown in Nason column 5, lines 45-63.

The examiner will now address the individual arguments and statements made by Appellant.

From pages 17 of the Appeal Brief, from the second paragraph, the Appellant argues that "Nason fails to provide teaching, suggestion or event motivation for a software component, which when executed, generates a first image dependent of code within the operating system during a first time, and generates a second image independent of code within the operating system during a second time, where the first and second images are substantially."

The examiner respectfully contends that Nason shows both a system that can display elements dependent on the native operating system (see column 5, lines 45-60) and further shows a complementary GUI display driver (see column 20, line 46 through column 21, line 28) and that this complementary GUI display is capable of being independent of the native operating system (see column 4, line 61 through column 5, line 4 and column 5, lines 45-63). The applications are shown to be combined with the kernel, which is further operating system independent. The existence of these two different application program interfaces (APIs) in the same system is a concept common to both the Nason and Fowler references. Where the fowler reference specifically shows a depiction of screen elements generated by the OS-dependent an OS-independent interfaces, with the elements being displayed in a manner where they appear very similar in appearance (see pages 4 and 6).

From pages 17 of the Appeal Brief, from the third paragraph, the Appellant argues that "Nason fails to disclose that, during the second time, the second images is adapted to overwrite the first image upon a display screen."

The examiner respectfully contends that as the applicant admits on page 17, paragraph 3 of the appeal brief, Nason teaches one image overwriting another image. Where these images, as proven above through the teachings of Nason, can be generated by any combination of platform independent or platform dependent APIs. Fowler further shows a case with two images (generated by different API) overlapping (see pages 4-6).

From pages 17 of the Appeal Brief, from the fifth paragraph, the Appellant argues that "Fowler cannot be combined with Nason to provide teaching or suggestion for the presently claimed computer readable storage device."

The examiner respectfully contends that Fowler teaches a system of mixing two APIs on the same display, similar to that of Nason (see page 1, paragraphs 1 and 2), and further teaches specifics of Java's interfaces AWT and Swing, where AWT uses heavyweight components (components that associate with native screen resources, and thus are dependent on the operating system), and where Swing uses lightweight components (components that borrow from screen resources of an ancestor, and thus are independent of the operating system) (see page 2, paragraphs 1 and 3), further similar to the alternate displays of Nason. Fowler further adds the teaching that the displayed components are similar in appearance (see pages 4-6).

From pages 18 of the Appeal Brief, from the third paragraph, the applicants' argue that Fowler teaches away from displaying an image with the AWT API during a first time, and displaying the same image with the Swing API during a second time.

In response, the examiner respectfully submits that though Fowler warns of problems that may be encountered when mixing AWT and Swing, Fowler teaches, on page 1, paragraph 2, the mixing of AWT and Swing in the same application program, where the teaching of display of the same image with a second API is taught by Nason, in column 25, lines 27-40, as shown supra. Fowler warns of problems in mixing but gives the case where mixing is necessary.

From page 18 of the Appeal Brief, from the fifth paragraph, the Appellant argues that "The Examiner has failed to adequately support and/or establish a prima facie ground of obviousness."

In response to applicant's argument that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, both references teach the use of a OS-independent interface and an OS-dependent interface used

together in the same system. Nason teaches the content and operating software can include JAVA, where Fowler teaches which API components in JAVA implement these OS-independent and an OS-dependent interfaces.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

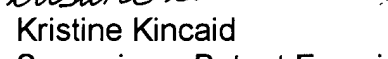
For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Conferees:

Dennis G. Bonshock
September 26, 2005

John W. Cabeca
Supervisory Patent Examiner
September 26, 2005



Kristine Kincaid
Supervisory Patent Examiner
September 26, 2005